Winter 2-1-1981

On the Complexity of Iterated Shuffle ; CU-CS-201-81

Manfred K. Warmuth
University of Colorado Boulder

David Haussler
University of Colorado Boulder

ON THE COMPLEXITY OF ITERATED SHUFFLE

by

Manfred K. Warmuth
David Haussler

Department of Computer Science
University of Colorado
Boulder, Colorado 80309

CU-CS-201-81 February, 1981

This research was supported in part by NSF Grant MCS 79-03838, the University of Colorado doctoral fellowship program, the Fulbright Commission of West Germany, and grants from Univac Corporation, and Storage Technology Corporation.

ABSTRACT

We demonstrate that the following problems are NP complete:

1) Given words w and w_1, w_2, ..., w_n, is w in the shuffle of w_1, w_2, ..., w_n?

2) Given words w and v, is w in the iterated shuffle of v?

From these results we show that the languages $\{\$w¢w^R : w \in \{a,b\}^*\}^{\circledast}$, $\bigcup_{w \in \{a,b,\}^*} (\$w)^{\circledast}$, $\{ab^n cde^n f : n \geq 0\}^{\circledast}$, and $\{a^{n+1} b^n c^n f^n : n \geq 0\}^{\circledast}$ are NP complete, where $\circledast$ denotes the iterated shuffle. By representing these languages in various ways using the operations shuffle, iterated shuffle, union, concatenation, intersection, intersection with a regular set, nonerasing homomorphisms, and inverse homomorphism, we obtain results on the complexity of language classes generated using various subsets of these operations. Finally, we show that the iterated shuffle of a regular set can be recognized in deterministic polynomial time.

I. INTRODUCTION

The operations of shuffle and iterated shuffle have been used by numerous researchers to describe sequential computation histories of concurrent programs [RID72], [SHA75], [KIM76], [SHA78], [ARA,KAG,TOK79]. Recently, there has been some investigation into the language generating power of the operations of shuffle and iterated shuffle, when used in combination with the more conventional operations of union, concatenation, intersection, intersection with a regular set, non erasing homomorphism and inverse homomorphism [OGD,RID78], [JAN79], [GIS79], [SLU80], [JAN81]. Jantzen [JAN79,81] has obtained numerous representations of the recursively enumerable languages and of the homomorphic images of the computation sequence sets of petri nets as the closure of various elementary language classes under certain combinations of the above operations, but allowing arbitrary homomorphisms or other forms of erasing via cancellation. On the other hand, it is clear that the class NP is closed under the fundamental set of operations listed above, in which we have omitted non erasing homomorphism and all forms of cancellation. Thus the language classes generated from the finite languages using these length preserving operations will be subsets of NP. Jantzen's results using non-erasing homomorphisms leave it unclear at which point, if any, the languages generated using these operations become intractable.

In this paper we will be concerned with distinguishing those language classes generated by various combinations of the above operations which are recognized in deterministic polynomial time from those that contain NP complete languages. Intuitively, the operations of shuffle and iterated shuffle are natural candidates for producing the kind of "hiding of information" which generates NP complete languages. This intuition is

borne out in what follows.

In Section II we give a small set of definitions. Then in Section III we present two basic NP complete problems. In the first, we are given a word w and words $w_1, w_2, \ldots, w_n$ and we ask whether w is in the shuffle of the words $w_1, w_2, \ldots, w_n$. This problem remains NP complete even if all words $w_1, w_2, \ldots, w_n$ are identical, which yields a second NP complete problem: for two words w and v, the problem of whether or not w is in the iterated shuffle of v is NP complete.

Using the above NP complete problems we show in Section IV that $\{\$w\¢w : w \in \{a,b\}^*\}^{\circledast}$, $\{\$w\¢w^R : w \in \{a,b\}^*\}^{\circledast}$, $\bigcup_{w \in \{a,b\}^*} (\$w)^{\circledast}$, $\{a\,b^n\,c\,d\,e^n\,f : n \geq 0\}^{\circledast}$ and $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq 0\}^{\circledast}$ are NP-complete, where $^{\circledast}$ denotes the iterated shuffle. Because of their simple structure, the last two languages can be expressed using various subsets of our fundamental set of operations. Hence we are able to use these languages to explore the complexity of the language classes generated by closing the finite sets under certain subsets of these basic operations. For instance, $\{a\,b^n\,c\,d\,e^n\,f : n \geq 0\}^{\circledast} = ((a\,c\,d\,f + b\,e)^{\circledast} \cap a\,b^*\,c\,d\,e^*\,f)^{\circledast}$, hence the closure of the finite languages under iterated shuffle and intersection with a regular set contains NP complete languages. Jantzen ([JAN 81]) has demonstrated that the above language is not accepted by any real time one-way multicounter machine. If $P \neq NP$, in fact this language is not accepted by any deterministic polynomial time algorithm. A summary of the complexity results of this type is given in Tables 4.1 and 5.1.

We conclude Section IV by exploring the effect of alphabet size on the complexity of the problems and languages we have considered. Basically, we show that two letters are enough to produce the degree of complexity achieved using larger alphabets.

Considering the language $\{a\,b^n\,c\,d\,e^n\,f : n \geq o\}^{\circledast}$ of Section IV, we see that the iterated shuffle of a linear language can be NP-complete. In Section V we give a deterministic polynomial time algorithm for the iterated shuffle of an arbitrary regular language. Using this result we show that the class *Shuf*, defined by Jantzen [JAN79,81] as the closure of the finite languages under the operations of union, shuffle and itereated shuffle, is contained in *P*. We also show that the closure of the finite languages under any pair of operations consisting of iterated shuffle and one operation in the set {intersection, non erasing homomorphism, inverse homomorphism and kleene closure} is contained in *P*. It remains open whether or not this is the case for the operations of iterated shuffle and concatenation. We suspect that in fact the class SE, obtained from *Shuf* by closing *Sluf* under concatenation and kleene closure, is contained in *P*. This class has been examined in several papers including [JAN81], [SHA78] and [KIM76], usually being presented as the class of languages obtained by extending the regular expressions to include the operations of shuffle and iterated shuffle. If, contrary to our hypothesis, this class is not in *P*, then perhaps there are NP complete languages which can be expressed by extending the regular expressions in this natural way.

In our final section, the complexity of the sets of computation histories resulting from the random concurrent execution of a set of processes is discussed in light of the results presented in the preceeding sections.

II. DEFINITIONS

Several different symbols for the operations shuffle and iterated shuffle have been used in the literature (see e.g. [RID72], [JAN79], [SHA78]). We use the symbol $\odot$ for shuffle and $\circledast$ for iterated shuffle, following Shaw ([SHA78]), because they correspond nicely to the operations concatenation $\cdot$ and kleene closure *. Formally, we have the following definitions: Given a finite alphabet Σ and $v, w \in \Sigma^*$ we define

$$v \odot w = \{v_1 w_1 v_2 w_2 \ldots v_k w_k : v_i, w_i \in \Sigma^* \text{ for } 1 \le i \le k, v = v_1 \ldots v_k \text{ and } w = w_1 \ldots w_k\},$$

$$w^{\textcircled{\odot}} = \{\lambda\},\ w^{\textcircled{i+1}} = w^{\textcircled{i}} \odot w \text{ and } w^{\circledast} = \bigcup_{i=0}^{\infty} w^{\textcircled{i}}.$$

Given words $w_1, w_2, \ldots, w_k \in \Sigma^*$, we define $\prod_{i=1}^{k} w_i = w_1 w_2 \ldots w_k$ and $\bigodot_{i=1}^{k} w_i = w_1 \odot w_2 \odot \ldots \odot w_k$. We will extend the above operations from strings to languages in the usual manner.

All the polynomial reductions of our paper are reductions from the following NP complete problem:

3-PARTITION:

GIVEN: a sequence of natural numbers $S = \langle n_i : 1 \le i \le 3m \rangle$ such that $B = (\sum_{i=1}^{3m} n_i)/m \in \mathbb{N}$ and for each i, $1 \le i \le 3m$, $\frac{B}{4} < n_i < \frac{B}{2}$.

QUESTION: Can S be partitioned into m disjoint subsequences $S_1, \ldots, S_m$ such that for each k, $1 \le k \le m$, S_k has exactly three elements and $\sum_{n \in S_k} n = B$. This was the first problem proven to be NP complete in the strong sense ([GAR,JOH 75, 80]). Thus it remains NP complete when the n_i are given in unary notation, which will be essential for our paper. The number B above will be referred to as the bound for S and the partition described above will be called a 3-partition.

III. TWO NP COMPLETE PROBLEMS

In this section we will demonstrate how the shuffle operation can be used to perform the partioning and addition required to determine if a given instance of 3-PARTITION has a solution. Our goal is to derive two fundamental NP complete problems involving the shuffle operation from 3-PARTITION. First, we present a few properties of the shuffle operation which will be used extensively in this paper.

Lemma 3.1: Given a finite alphabet Σ containing the symbols a, b, and c, the words v, x, w, w_1, w_2, ..., $w_\ell \in \Sigma^*$, and natural numbers n, n_1, n_2, ..., n_ℓ, for some $\ell > 0$, where x does not begin with a, b or c and w_i does not begin with c for $i : 1 \le i \le \ell$, then

i) $a^k b^n c^k w \in x \odot \bigcirc\!\!\!\!\!\!\pi_{i=1}^{\ell} a b^{n_i} c w_i \Leftrightarrow$ there exist distinct indicies $j_1, \ldots, j_k$ such that $n = \sum_{r=1}^{k} n_{j_r}$ and $w \in x \odot \left(\bigcirc\!\!\!\!\!\!\pi_{r=1}^{k} w_{j_r} \right) \odot \bigcirc\!\!\!\!\!\!\pi_{\substack{i=1 \\ i \notin \{j_1,\ldots,j_k\}}}^{\ell} a b^{n_i} c w_i$

ii) $a^n c^n w \in x \odot \bigcirc\!\!\!\!\!\!\pi_{i=1}^{\ell} a^{n_i} c^{n_i} w_i \Leftrightarrow$ there exist distinct indicies $j_1, \ldots, j_k$ such that $n = \sum_{r=1}^{k} n_{j_r}$ and $w \in x \odot \left(\bigcirc\!\!\!\!\!\!\pi_{r=1}^{k} w_{j_r} \right) \odot \bigcirc\!\!\!\!\!\!\pi_{\substack{i=1 \\ i \notin \{j_1,\ldots,j_k\}}}^{\ell} a^{n_i} c^{n_i} w_i$

iii) $a^{n+1} c^n w \in x \odot \bigcirc\!\!\!\!\!\!\pi_{i=1}^{\ell} a^{n_i+1} c^{n_i} w_i \Leftrightarrow$

there exists an index k such that $n_k = n$ and $w \in x \odot w_k \odot \bigcirc\!\!\!\!\!\!\pi_{\substack{i=1 \\ i \neq k}}^{\ell} a^{n_i+1} c^{n_i} w_i$

Proof: i) The words $w_1, w_2, \ldots, w_\ell$ don't start with the symbol c. Therefore, to account for the a's and c's, the prefix $a^k b^n c^k w$ must be formed by shuffling exactly k prefixes of the $a b^{n_i} c w_i$'s of the form $a b^{n_i} c$ and the result follows.

ii) Since the w_i's don't start with c, the prefix $a^n c^n$ of $a^n c^n w$ has to be created by shuffling prefixes of the $a^{n_i} c^{n_i} w_i$'s of the form $a^p c^q$, $p \geq q \geq 0$. Since n a's are needed followed by an equal amount of c's only prefixes of the form $a^p c^p$ can be used and the result follows.

iii) To get the prefix $a^{n+1} c^n$ of $a^{n+1} c^n w$, prefixes of the $a^{n_i+1} c^{n_i} w_i$'s of the form $a^p c^q$, $p > q \geq 0$, have to be shuffled. Since there is only one more a than c in $a^{n+1} c^n$ at most one prefix can be used. □

Lemma 3.2: Let $S = \langle n_i : 1 \leq i \leq 3m \rangle$ be an instance of 3-PARTITION with bound B. The following are equivalent:

i) S has a 3-partition,

ii) $(a^3 b^B c^3)^m \in \textcircled{\pi}_{i=1}^{3m} a b^{n_i} c$,

iii) $(a^B b^B)^m \in \textcircled{\pi}_{i=1}^{3m} a^{n_i} b^{n_i}$.

Proof: It is easily seen that i $\Rightarrow$ ii and i $\Rightarrow$ iii. We simply merge using the shuffle operation each of the m triplets guaranteed by the partition to obtain the desired word. To see that ii $\Rightarrow$ i we argue by induction on m. The case m = 1 is trivial. If $w = a^3 b^B c^3 (a^3 b^B c^3)^m \in \textcircled{\pi}_{i=1}^{3(m+1)} a b^{n_i} c$ for some $S = \langle n_i : 1 \leq i \leq 3(m+1) \rangle$ an instance of 3-PARTITION, with bound B, then the initial $a^3 b^B c^3$ of w must be the result of shuffling exactly three words from $\langle a b^{n_i} c : 1 \leq i \leq 3(m+1) \rangle$. This follows directly from Lemma 3.1 part i. Thus the remaining suffix $(a^3 b^B c^3)^m$ of w is formed by shuffling the remaining $3m$ $a b^{n_i} c$'s. By hypothesis then, these remaining n_i's have a 3-partition. Hence the entire sequence S has a 3-partition and the

inductive step is complete. The proof that iii$\Rightarrow$i is similar. Assuming that $a^B b^B (a^B b^B)^m \in \bigcirc\!\!\!\!\pi_{i=1}^{3(m+1)} a^{n_i} b^{n_i}$ for S, B as above, again the initial $a^B b^B$ must be obtained from shuffling some words from $T = \langle a^{n_i} b^{n_i} : 1 \leq i \leq 3(m+1) \rangle$. Here we use Lemma 3.1 part ii. The fact that in any instance of 3 PARTITION we require that $\frac{B}{4} < m_i < \frac{B}{2}$ for all i implies that we must use exactly three words from T. Our result then follows by induction as above. □

As an immediate consequence of the above lemma, we have the following theorem.

Theorem 3.1: Given a finite alphabet $\Sigma = \{a,b\}$ and words w, w_1, ..., $w_k \in \Sigma^*$, the problem of whether or not $w \in \bigcirc\!\!\!\!\pi_{i=1}^{k} w_i$ is NP complete.

Proof: We use Lemma 3.2 to encode an instance of 3 PARTITION into the above problem. Since 3 PARTITION is strongly NP complete, we can assume the numbers are given in unary notation, thus the transformation will be polynomial. □

The above problem remains NP complete when we restrict ourselves to the case in which all of the w_i's are identical. To demonstrate this, we will use the set $\bigcirc\!\!\!\!\pi_{k=1}^{3m} (\pi_{i=1}^{3m} a b^{n_i} c)$ in place of the simpler set $\bigcirc\!\!\!\!\pi_{i=1}^{3m} a b^{n_i} c$ used in Lemma 3.1. The essence of our argument is contained in the following lemma.

Lemma 3.3: Given a sequence of words $T = \langle a b^{n_i} c : 1 \leq i \leq 3m \rangle$ let $u_i = \pi_{k=1}^{i} a b^{n_k} c$ and $v_i = \pi_{k=i}^{3m} a b^{n_k} c$ for $1 \leq i \leq 3m$.

Let $w = u_{3m} = v_1 = \pi_{i=1}^{3m} a b^{n_i} c$

Let $p = \pi_{i=1}^{3m-1} u_i$ and $q = \pi_{i=2}^{3m} v_i$.

Then for any B, $p(a^3 b^B c^3)^m q \in w^{(3m)}$ iff $(a^3 b^B c^3)^m \in \bigcirc\!\!\!\!\pi_{i=1}^{3m} a b^{n_i} c$.

Proof: The "if" part of this proof is very simple. We form the p and q of $p(a^3 b^B c^3)^m q$ by collecting the proper prefixes and suffixes of each of the copies of w, in each case leaving one string of the form $a b^{n_i} c$ for a different i to contribute to $(a^3 b^B c^3)^m$. The sequence of middle words obtained in this way will be T, which by hypothesis will shuffle to form the desired word $(a^3 b^B c^3)^m$. To see that the "only if" part holds, we notice that every subword of $p(a^3 b^B c^3)^m q$ the form $a^k b^* c^k$ for $k \in \{1,3\}$ must have come from exactly k subwords of the form $a b^* c$, each from a different copy of w. This follows from Lemma 3.1 part i. Since $p(a^3 b^B c^3)^m q \in w^{(3m)}$, a total of 3m copies of each $a b^{n_i} c$ are involved. p and q consume exactly 3m-1 copies of each $a b^{n_i} c$ for $1 \le i \le 3m$, leaving exactly one copy of each $a b^{n_i} c$ to be shuffled together to form the middle word $(a^3 b^B c^3)^m$. Thus $(a^3 b^B c^3)^m \in \bigcirc\!\!\!\!\pi_{i=1}^{3m} a b^{n_i} c$. □

Using this lemma, we obtain a second NP complete problem.

Theorem 3.2: Given a finite alphabet $\Sigma = \{a,b,c\}$ and words $w, v \in \Sigma^*$, the problem of whether or not $v \in w^{\circledast}$ is NP complete.

Proof: Using Lemmas 3.2 and 3.3, given an instance of 3-PARTITION $S = \{n_i : 1 \le i \le 3m\}$ with bound B we can find words w and v such that $v \in w^{(3m)}$ iff S has a 3-partition. Thus we can reduce 3-PARTITION to the above problem. □

IV. NP COMPLETE LANGUAGES AND THEIR REPRESENTATIONS

Each of the NP complete problems from Section III had the form: given a word and a simple language, is the word contained in the language? In this section we will exhibit fixed languages that contain within them encodings of each of the instances of one of the problems from the previous section. Thus we will convert the problem of membership in which both the word and the language are variable, into the problem of membership in a fixed language. Our basic technique is demonstrated in our first lemma.

Lemma 4.1: Given words v and w over some finite alphabet Σ not including the symbols \$ and ¢, the following are equivalent:

i) $w \in v^{\textcircled{k}}$

ii) $(\$ v ¢)^k w \in \{\$ x ¢ x : x \in \Sigma^*\}^{\circledast}$

iii) $(\$ v ¢)^k w^R \in \{\$ x ¢ x^R : x \in \Sigma^*\}^{\circledast}$

(where w^R indicates the reverse of the word w)

iv) $\$ v \$^k w \in \bigcup_{x \in \Sigma} (\$ x)^{\circledast}$ and $|w| = k|v|$

Proof: Obviously i $\Rightarrow$ ii, we simply shuffle k copies of \$ v ¢ v by collecting the prefixes \$ v ¢ in the front and shuffling the suffixes v to form w. To show that ii $\Rightarrow$ i we argue that the prefix $(\$ v ¢)^k$ of $(\$ v ¢)^k w$ must have been formed by shuffling k prefixes of the form \$ v ¢ from words in $\{\$ x ¢ x : x \in \Sigma^*\}$. This follows from the fact that v does not contain the symbols \$ and ¢. Hence the suffix w of $(\$ v ¢)^k w$ must have been formed by shuffling k copies of the suffix v.

The proof that i $\Leftrightarrow$ iii is very similar to i $\Leftrightarrow$ ii. i $\Rightarrow$ iv is obvious, since if $w \in v^{\textcircled{k}}$, $\$ v \$^k w \in (\$ v)^{\textcircled{*}}$. To show that iv $\Rightarrow$ i we note that if $\$ v \$^k w \in (\$ x)^{\circledast}$ for some x, then the prefix \$ v of $\$ v \$^k w$ must be a prefix of \$x. Further, $\$ v \$^k w$ has k + 1 \$'s, hence $\$ v \$^k w \in (\$ x)^{\textcircled{k+1}}$.

Thus since $|w| = k|v|$, v must equal x, and w must arise from shuffling the remaining k copies of v. □

Theorem 4.1: The languages $\{\$w\¢w : w \in \{a,b,c\}^*\}^{\circledast}$, $\{\$w\¢w^R : w \in \{a,b,c\}^*\}^{\circledast}$, and $\bigcup_{x \in \{a,b,c\}^*} (\$x)^{\circledast}$ are NP complete.

Proof: Follows from Theorem 3.2 and Lemma 4.1. □

In our next theorem we will employ our techniques for converting a variable language membership problem into a fixed language membership problem to the problems considered in Lemma 3.2.

Theorem 4.2: The languages $\{a\,b^n\,c\,d\,e^n\,f : n \geq 0\}^{\circledast}$ and $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq 0\}^{\circledast}$ are NP complete.

Proof: We claim that given natural numbers B, n_i, ..., n_{3n} the following hold:

1) $(d^3\,e^B\,f^3)^m \in \bigcirc\!\!\!\!\!\!\pi_{i=1}^{3m}\; d\,e^{n_i}\,f \iff$

$$w_1 = \left(\prod_{i=1}^{3m} a\,b^{n_i}\,c\right)(d^3\,e^B\,f^3)^m \in \{a\,b^n\,c\,d\,e^n\,f : n \geq 0\}^{\circledast}.$$

2) $(c^B\,d^B)^m \in \bigcirc\!\!\!\!\!\!\pi_{i=1}^{3m}\; c^{n_i}\,d^{n_i} \iff$

$$w_2 = \left(\prod_{i=1}^{3m} a^{n_i+1}\,b^{n_i}\right)(c^B\,d^B)^m \in \{a^{n+1}\,b^n\,c^n\,d^n : n \geq 0\}^{\circledast}.$$

The forward implications are obvious, we need only select the sets $L_1 = \{a\,b^{n_i}\,c\,d\,e^{n_i}\,f : 1 \leq i \leq 3m\}$ and $L_2 = \{a^{n_i+1}\,b^{n_i}\,c^{n_i}\,d^{n_i} : 1 \leq i \leq 3m\}$ from $\{a\,b^n\,c\,d\,e^n\,f : n \geq 0\}$ and $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq 0\}$ and shuffle them together as proscribed in w_1 and w_2, respectively. To prove the reverse implication for (1), assume that $w_1 \in \bigcirc\!\!\!\!\!\!\pi_{i=1}^{\ell}\; a\,b^{k_i}\,c\,d\,e^{k_i}\,f$ for some k_1, ..., $k_\ell \in N$. Since w_1 contains 3m a's, $\ell = 3m$. Using Lemma 3.1 part (i) iteratively, we can show that $\langle k_1,\ldots,k_\ell\rangle$ is a permutation of $\langle n_1,\ldots,n_{3m}\rangle$ and $(d^3e^Bf^3)^m \in \bigcirc\!\!\!\!\!\!\pi_{i=1}^{3m}\; d\,e^{k_i}\,f$,

which establishes the result. The reverse implication for (2) is proved in a similar manner using Lemma 3.1 part (iii).

Now using this claim along with Lemma 3.2, we see that we can reduce an instance of 3-PARTITION to a question of membership in the language $\{a\,b^n\,c\,d\,e^n\,f : n \geq o\}^{\circledast}$ or $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq o\}^{\circledast}$. Hence these languages are NP complete. □

The language $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq o\}^{\circledast}$ is an interesting borderline case. The extra "a" provides the minimal amount of asymetry needed to determine the number of words from $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq o\}$ shuffled in forming a word from $\{a^{n+1}\,b^n\,c^n\,d^n : n \geq o\}^{\circledast}$. We could easily replace the extra "a" with a special marker \$, i.e., the language $\{\$a^n\,b^n\,c^n\,d^n : n \geq o\}^{\circledast}$ is also NP complete. On the other hand, the corresponding symetric language $\{a^n\,b^n\,c^n d^n : n \geq 1\}^{\circledast}$ reduces to $(a\,b\,c\,d)^{\circledast}$, which we will show in Section V can be recognized in deterministic polynomial time.

Owing to their simple structure, the languages from Theorem 4.2 are of central importance in obtaining results on the power of the operation of iterated shuffle, when used in conjunction with other basic language operations. We will consider the complexity of various classes of languages generated by taking the closure of the finite languages under subsets of the operations shuffle, iterated shuffle, union, concatenation, intersection, intersection with a regular set, non erasing homomorphism, and inverse homomorphism, denoted $\odot$, $\circledast$, $+$, $\cdot$, $\cap$, $\cap R$, h, h^{-1} respectively. Thus all our language classes will be contained in NP. On the other hand, without iterated shuffle the languages generated are always regular, since the regular languages are closed under all of the remaining operations.

Our next theorem gives numerous sets of operations which can be used to generate NP complete languages from finite sets. The examples of NP complete languages given will include some of those given above, along with a few variations on these langugaes, including $\{\$(a\bar{a})^n(b\bar{b})^n(c\bar{c})^n(d\bar{d})^n : n \geq 1\}^{\circledast}$, $\{a\,b^n c\,\$^n\,d\,e^n\,f : n \geq 0\}^{\circledast}$ and $\{\bar{a}ab^n(cd)^n e^n f\bar{f}\}^{\circledast} \cap \bar{a}^*(ab^*c)^*(dc)^*(d^3ef^3)^*\bar{f}^*$. It is easily demonstrated that these languages are NP complete using the techniques from Theorem 4.2. Before giving the theorem, we prove a few useful technical lemmas.

Lemma 4.2: For any $k \geq 1$, $(a_1\bar{a}_1 \ldots a_k\bar{a}_k)^{\circledast} \cap a_1\bar{a}_1 \ldots a_k\bar{a}_k \odot (\bar{a}_1a_1 \ldots \bar{a}_ka_k)^{\circledast} = \{(a_1\bar{a}_1)^n \ldots (a_k\bar{a}_k)^n : n \geq 1\}$.

Proof: This result follows from a simple counting argument involving the ratios of barred and unbarred letters in any word in the given intersection. The essence of the argument is given in the proof of Corollary 3.1 part f of [JAN 81], so we will not repeat it here. □

Lemma 4.3: Let $\Sigma = \{a_1, \ldots, a_{2k+1}\}$ for some $k > 0$ and let $h : \Sigma^* \to \bar{\Sigma}^*$ be the homomorphism defined by $h(a_1) = \bar{a}_1$, $h(a_{2k+1}) = \bar{a}_{2k}$, $h(a_{2i}) = \bar{a}_{2i}\bar{a}_{2i-1}$ and $h(a_{2i+1}) = \bar{a}_{2i}\bar{a}_{2i+1}$ for $i : 1 \leq i < k$. Let $L_1 = h^{-1}((\bar{a}_1 \ldots \bar{a}_{2k})^{\circledast}) \cap a_1(\Sigma - \{a_1\})^*$ and let $L_2 = \{a_1a_2^n a_3 \ldots a_{2k-1}a_{2k}^n a_{2k+1} : n \geq 0\}$. Then $L_1 = L_2$.

Proof: First, assume that we are given $w = a_1a_2^n a_3 \ldots a_{2k-1}a_{2k}^n a_{2k+1} \in L_2$. $h(w) = \bar{a}_1(\bar{a}_2\bar{a}_1)^n(\bar{a}_2\bar{a}_3) \ldots (\bar{a}_{2k-2}\bar{a}_{2k-1})(\bar{a}_{2k}\bar{a}_{2k-1})^n a_{2k} = (\bar{a}_1\bar{a}_2)^{n+1} \ldots (\bar{a}_{2k-1}\bar{a}_{2k})^{n+1} \in (\bar{a}_1 \ldots \bar{a}_{2k})^{\circledast}$. Hence $w \in h^{-1}((\bar{a}_1 \ldots \bar{a}_{2k})^{\circledast})$. Obviously, $w \in a_1(\Sigma - \{a_1\})^*$, hence $w \in L_1$. Thus $L_2 \subseteq L_1$.

Now, assume that we are given $w = a_1 x \in L_1$. Since $h(a_1) = \bar{a}_1$, $h(x) \in \bar{a}_2 \ldots \bar{a}_{2k} \odot \bigodot_{i=1}^{n} \bar{a}_1 \ldots \bar{a}_{2k}$ for some $n \geq 0$. a_2 and a_3 are the only letters in $\Sigma - \{a_1\}$ whose images under h begin with $\bar{a}_1$ or $\bar{a}_2$. Since $h(a_2) =$

$\bar{a}_2\bar{a}_1$ and $h(a_3) = \bar{a}_2\bar{a}_3$, there must exist $m : 0 \le m \le n$ such that $x = a_2^m a_3 y$, where $h(y) \in \bar{a}_4 \ldots \bar{a}_{2k} \odot \overset{m}{\underset{i=1}{\circledcirc}} \bar{a}_3 \ldots \bar{a}_{2k} \odot \overset{n-m}{\underset{i=1}{\circledcirc}} \bar{a}_1 \ldots \bar{a}_{2k}$. This is only possible if $n - m = 0$, i.e., if $x = a_2^n a_3 y$ and $h(y) \in \bar{a}_4 \ldots \bar{a}_{2k} \odot \overset{n}{\underset{i=1}{\circledcirc}} \bar{a}_3 \ldots \bar{a}_{2k}$. In this case a_4 and a_5 are the only letters in $\Sigma - \{a_1\}$ whose images under h begin with $\bar{a}_3$ or $\bar{a}_4$. Hence we can repeat the above argument. Continuing in this manner, it is apparent that $x = a_2^n a_3 \ldots a_{2k}^n a_{2k+1}$, and thus $w \in L_2$. Hence $L_1 = L_2$. □

Lemma 4.4: Let $\Delta = \Sigma \cup \{\$\}$ where $\$ \notin \Sigma$. Then for any $L \subseteq \Delta^*$, if $(L \cap \$\Sigma^*)^{\circledast}$ is NP complete, then there exists an alphabet Γ and a non erasing homomorphism $h : \Delta^* \to \Gamma^*$ such that $(h(L \cap \Delta^* - \Sigma^+))^{\circledast}$ is NP complete.

Proof: Let $\Gamma = \Delta \cup \{\cent\}$ where $\cent \notin \Delta$. Let h be the homomorphism defined by $h(\$) = \cent\,\$$ and $h(a) = a$ for $a \in \Sigma$. Let $T = (h(L \cap \Delta^* - \Sigma^+))^{\circledast} \cap \cent^*\Delta^*$. It is easily verified that $T = \bigcup_{m=0}^{\infty} \cent^m (L \cap \$\Sigma^*)^{\circledcirc}$.

Now assume that there is a deterministic polynomial time algorithm for T. Since $w \in (L \cap \$\Sigma^*)^{\circledast}$ if and only if $\cent^m w \in T$ for some $n : 0 \le n \le |w|$, we can easily convert this algorithm into a deterministic polynomial time algorithm for $(L \cap \$\Sigma^*)^{\circledast}$. Since this language is NP complete, it follows that T is NP hard. Since T is obviously in the class NP, it follows that T is NP complete. Finally, since T is NP complete, it follows that $(h(L \cap \Delta^* - \Sigma^+))^{\circledast}$ is NP complete. □

Theorem 4.3: There exists a finite alphabet Σ such that the closure of the class of finite languages in Σ^* under each of the following sets of operations contains NP complete languages:

(1) $\{\circledast, \cap R\}$,

(2) any set in $\{\circledast\} \times \{\odot, \cdot\} \times \{\cap, h, h^{-1}\}$

or $\{\circledast\} \times \{\cap\} \times \{h, h^{-1}\}$

and $\{\circledast, h, h^{-1}\}$.

Proof: ad.1. This follows from the fact that $((\$+abcd)^{\circledast} \cap \$a^*b^*c^*d^*)^{\circledast} = \{\$a^nb^nc^nd^n : n \geq 0\}^{\circledast}$. (See Theorem 4.2 and comments following).

ad.2. For each of the nine sets of operations listed we give an example of an NP complete language generated from finite sets using these operations. Detailed verification of the examples is left to the reader.

i) $\{\circledast, \cdot, \cap\}$

$(a \cdot (abcd)^{\circledast} \cap a^{\circledast} \cdot b^{\circledast} \cdot c^{\circledast} \cdot d^{\circledast})^{\circledast} = \{a^{n+1}b^nc^nd^n : n \geq 0\}^{\circledast}$.

ii) $\{\circledast, \odot, \cap\}$

$((\$\odot(a\bar{a}b\bar{b}c\bar{c}d\bar{d})^{\circledast}) \cap (\$a\bar{a}b\bar{b}c\bar{c}d\bar{d}\odot(\bar{a}a\bar{b}b\bar{c}c\bar{d}d)^{\circledast}))^{\circledast} =$

$\{\$(a\bar{a})^n(b\bar{b})^n(c\bar{c})^n(d\bar{d})^n : n \geq 1\}^{\circledast}$. This follows using Lemma 4.2.

iii) $\{\circledast\} \times \{\odot, \cdot\} \times \{h^{-1}\}$

Let $\Sigma = \{\$, a, b, c, d, e, f\}$ and let $h : \Sigma^* \to \bar{\Sigma}^*$ be the homomorphism defined by $h(a) = \overline{\$a}$, $h(b) = \overline{ba}$, $h(c) = \overline{bc}$, $h(\$) = \overline{dc}$, $h(d) = \overline{de}$, $h(e) = \overline{fe}$ and $h(f) = \bar{f}$. Let $\square$ be an operation in $\{\odot, \cdot\}$. Then $(h^{-1}(\$\square(\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}))^{\circledast} = \{ab^nc\$^nde^nf : n \geq 0\}^{\circledast}$, using Lemma 4.3.

iv) $\{\circledast, h^{-1}, \cap\}$

Let Σ and h be defined as in (iii), except let $h(a) = \bar{a}$. Let $g : \Sigma^* \to \bar{\Sigma}^*$ be the homomorphism defined by $g(a) = \bar{a}$, $g(x) = \lambda$ for $x \in \Sigma - \{a\}$. Then $(h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap g^{-1}(\bar{a}))^{\circledast} = \{ab^nc\$^nde^nf : n \geq 0\}^{\circledast}$, again using Lemma 4.3.

v) $\{\circledast, h, h^{-1}\}$

Let Σ and h be defined as in (iv). Using Lemma 4.3, it follows that $(h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap a(\Sigma - \{a\})^*)^{\circledast} = \{ab^nc\$^nde^nf : n \geq 0\}^{\circledast}$ which is NP complete.

Hence by Lemma 4.4 there exists a non erasing homomorphism g such that $(g(h^{-1}((\overline{a}\,\overline{b}\,\overline{c}\,\overline{d}\,\overline{e}\,\overline{f})^{\circledast}) \cap \Sigma^* - (\Sigma - \{a\})^+))^{\circledast}$ is NP complete. However, $h^{-1}((\overline{a}\,\overline{b}\,\overline{c}\,\overline{d}\,\overline{e}\,\overline{f})^{\circledast}) \subseteq \Sigma^* - (\Sigma - \{a\})^+$, hence $(g(h^{-1}((\overline{a}\,\overline{b}\,\overline{c}\,\overline{d}\,\overline{e}\,\overline{f})^{\circledast})))^{\circledast}$ is NP complete.

vi) $\{\circledast, h, \cap\}$. Let $\Sigma = \{a, b, c, d\}$ and $\Delta = \Sigma \cup \{\$\}$.
Let $L = (\$ + a\,\overline{a}\,b\,\overline{b}\,c\,\overline{c}\,d\,\overline{d})^{\circledast} \cap (\$\,a\,\overline{a}\,b\,\overline{b}\,c\,\overline{c}\,d\,\overline{d} + \overline{a}\,a\,\overline{b}\,b\,\overline{c}\,c\,\overline{d}\,d)^{\circledast}$. Using Lemma 4.2, it is easily verified that $(L \cap \$\Sigma^*)^{\circledast} =$ $\{\$(a\,\overline{a})^n (b\,\overline{b})^n (c\,\overline{c})^n (d\,\overline{d})^n : n \geq 1\}^{\circledast}$, which is NP complete. Hence by Lemma 4.4 there exists a non erasing homomorphism h such that $(h(L \cap \Delta^* - \Sigma^+))^{\circledast}$ is NP complete. However $L \subseteq \Delta^* - \Sigma^+$, hence $(h(L))^{\circledast}$ is NP complete.

vii) $\{\circledast\} \times \{\odot, \cdot\} \times \{h\}$
Let $\Sigma = \{a, b, c, d, e, f\}$ and let $h : \Sigma^* \to (\Sigma \cup \overline{\Sigma})^+$ be the homorphism defined by $h(a) = \overline{a}\,a$, $h(b) = b$, $h(c) = c\,d$, $h(d) = d$, $h(e) = e$, $h(f) = f\,\overline{f}$.
Let $R = \overline{a}^*(a\,b^*\,c)^*(d\,c)^*(d^3\,e^*\,f^3)^*\,\overline{f}^*$. Let $\square$ be an operation in $\{\odot, \cdot\}$.
Let $L = (h(a \square (b\,c\,e)^{\circledast} \square f))^{\circledast}$. It is easily verified that
$L \cap R = \{\overline{a}\,a\,b^n (c\,d)^n e^n f\,\overline{f} : n \geq 0\}^{\circledast} \cap R$, which is NP complete.

Hence L is NP complete.

This completes the proof of Theorem 4.3. □

The results of the second part of Theorem 4.3 are summarized in Table 4.1. For each pair of distinct operations in the set $\{\cdot, \odot, \cap, h, h^{-1}\}$ an indication of the complexity of languages generated using these operations in conjunction with iterated shuffle is given. A "c" in the row and column for operations O_1 and O_2 indicates that the closure of the finite sets under the operations $\circledast$, O_1 and O_2 contains NP complete languages.

operations	$\cdot$	$\cap$	h^{-1}	h
$\odot$	?	c	c	c
h	c	c	c	
h^{-1}	c	c		
$\cap$	c			

Table 4.1

In most of our NP completeness results up to this point, we have relied on alphabets of size three or larger. It is obvious that with the alphabet $\Sigma = \{a\}$, Theorems 3.1 and 3.2 do not hold, and that all languages generated from a^* using the operations considered above, excluding h and h^{-1}, are regular. The case in which Σ has just two elements remains unexplored. The possibility remains that there is some sort of complexity gap between alphabets of size two and three in the context of some of our iterated shuffle results. The following indicates that this is not the case.

Lemma 4.5: Given the alphabet $\Delta = \{a_1, \ldots a_k\}$ we define the homomorphism $h : \Delta^* \to \{a\,b\}^*$ as $h(a_i) = a^{i+1}\,b^i$. Then for any languages L_1, $L_2 \subset \Delta^*$ we have

i) $w \in L_1^{\circledast}$ iff $h(w) \in (h(L_1))^{\circledast}$

ii) $w \in L_1 \odot L_2$ iff $h(w) \in h(L_1) \odot h(L_2)$

iii) $w \in L_1 \cdot L_2$ iff $h(w) \in h(L_1) \cdot h(L_2)$

iv) $w \in L_1 + L_2$ iff $h(w) \in h(L_1) + h(L_2)$

v) $w \in L_1 \cap L_2$ iff $h(w) \in h(L_1) \cap h(L_2)$

Proof: iii, iv and v follow from the fact that h is a code, i.e., the function $h : \Delta^* \to \{a,b\}^*$ is injective. For parts i and ii, it is obvious that if $w \in L_1^{\circledast}$ then $h(w) \in (h(L_1))^{\circledast}$ and if $w \in L_1 \odot L_2$ then $h(w) \in h(L_1) \odot h(L_2)$, we need only shuffle images of letters in Δ as units. For the reverse implications we use Lemma 3.1 part iii iteratively to "decode" a word in the range of h formed by shuffling words from the range of h. □

The "shuffle resistant" code defined in the above theorem can be used to reduce the size of the alphabet required for our results.

Corollary 1: Theorems 3.2, 4.1 and the parts of 4.3 not involving the operations h and h^{-1} hold for $\Sigma = \{a,b\}$.

Proof: Follows directly from Lemma 4.5. □

V. LANGUAGE CLASSES CONTAINED IN P

In Section IV, we have presented a variety of NP complete languages, each of which had the form $L^{\circledast}$ for some language L. In each case there is a strong structual relationship between the language L and the language $\{ww : w \in \Sigma^*\}$. Each of the languages exhibits some kind of unbounded pairing or repetition along with markers which in the simplest case reduces to counting as in $\{a\,b^n\,c\,d\,e^n\,f : n \geq 1\}$. Hence there are NP complete languages among the iterated shuffles of linear languages. We have not however, been able to give an NP complete language of form $R^{\circledast}$ for regualar R. The next theorem indicates that this is impossible unless P = NP.

Theorem 5.1: For any regular R, the language $R^{\circledast}$ can be recognized in deterministic polynomial time.

Proof: Let us suppose we are given a finite automaton A for the Language R with k states, possibly nondeterministic, but without λ-moves. To recognize a word $w \in R^{\circledast}$, we imagine that we begin with a pile of pebbles placed in the start state of A. Each time we read a letter ℓ from w, we must find a state q of A which has one or more pebbles in it, such that q maps with letter ℓ to some state q' in the automaton A. Upon finding such a state q, we remove a pebble from q and place it in q'. We claim that $w \in R^{\circledast}$ iff there is a way to move some initial pile of pebbles while reading w such that when we are finished, all the pebbles are in a final state. This follows since the movement of each pebble contributes one word from R, which is shuffled with the words obtained from moving the other pebbles.

We will determine if $w \in R^{\circledast}$, for $w \neq \lambda$, by computing all possible final pebble configurations for w. If $|w| = n$ then we need to start with at most n pebbles in the start state. At any given time as we are reading

a letter of w, there are at most n pebbles in any given state. Since A has k states, the number of possible intermediate pebblings of the automaton A is less than n^k. Hence, we can keep track of all possible pebble configurations of A as we read w. Each time we read a letter from w we revise our list of possible configurations using $O(n^k)$ time. Thus our total time is $O(n \cdot n^k) = O(n^{k+1})$, which completes the proof. □

Corollary 5.1: For any finite alphabet, the closure of the finite languages in Σ^* under any set of operations in $\{\circledast\} \times \{*, \cap, h, h^{-1}\}$ is contained in P, the class of all languages recognized in deterministic polynomial time.

Proof: The following calculation rules are easily verified. For any languages $L, L_1, \ldots, L_k \subseteq \Sigma^*$ and homomorphisms $h_1 : \Sigma^* \to \Delta^*$ and $h_2 : \Delta^* \to \Sigma^*$,

(i) $(L^*)^{\circledast} = (L^{\circledast})^* = (L^{\circledast})^{\circledast} = L^{\circledast}$,

(ii) $(\bigcap_{i=1}^{k} L_i^{\circledast})^{\circledast} = \bigcap_{i=1}^{k} L_i^{\circledast}$,

(iii) $(h_2^{-1}(L^{\circledast}))^{\circledast} = h_2^{-1}(L^{\circledast})$ and

(iv) $(h_1(L^{\circledast}))^{\circledast} = (h_1(L))^{\circledast}$.

Using these rules, we can easily find a representation for any language in any of the above classes in which the applications of the operation $\circledast$ do not appear nested. Hence the result follows from Theorem 5.1, using the elementary closure properties of the class P. □

Following Jantzen [JAN81], we make the following definitions:

Definition: Given a finite alphabet Σ, we define the language class Shuf_Σ as the closure of the finite languages over Σ under the operations $+, \odot$ and $\circledast$ and the language class SE_Σ as the closure of the finite languages over Σ under the operations $+, \cdot, *, \odot$ and $\circledast$.

The class SE_Σ is the class of languages definable by the shuffle expressions of [SHA 78] or the c-expressions of [KIM 76] over the alphabet Σ. In [JAN 81] Jantzen shows that $Shuf_\Sigma \subsetneq SE_\Sigma$ and he gives a representation for an arbitrary $L \in Shuf_\Sigma$ as $\bigcup_{i=1}^{m} M_i \odot N_i^{\circledast}$ for finite $M_i, N_i \subset \Sigma^*$. A language of the form $R \odot S^{\circledast}$ for regular sets R and S can be recognized in deterministic polynomial time by a simple extension of the algorithm of the previous theorem. Thus we have the following corollary.

Corollary 5.2: For any finite alphabet Σ, the class $Shuf_\Sigma$ is contained in *P*.

Table 5.1 summarizes the state of our knowledge about the complexity of languages generated by the operation of iterated shuffle in conjunction with any other single basic language operation. A "P" in the row for operation $\square$ indicates that the closure of the finite languages in Σ^* under the operations $\circledast$ and $\square$ is contained within *P* for any Σ. A "C" indicates that there exists a finite Σ such that the closure of the finite languages in Σ^* under $\circledast$ and $\square$ contains NP complete languages.

OPERATION	COMPLEXITY
+	P
·	?
*	P
$\cap$	P
h	P
h^{-1}	P
$\odot$	P
$\cap R$	C

Table 5.1

The complexity of the class SE_Σ for $|\Sigma| \geq 2$ remains open, as does the complexity of any of the classes of languages generated by closure of the finite languages under any subset of the operations $+, \cdot, *, \odot$ and $\circledast$ which includes both $\cdot$ and $\circledast$. We have not found any NP complete languages in the class SE_Σ for any Σ, nor have we been able to exhibit deterministic polynomial time algorithms for any class of languages containing the finite languages, and closed under $\cdot$ and $\circledast$. Our hypothesis is that such polynomial time algorithms can be found, and indeed that SE_Σ is contained in P for any Σ.

VI. APPLICATIONS TO THE STUDY OF CONCURRENT COMPUTATION

A natural problem to consider in the area of concurrent computation is the following: Can a given sequence of actions occur in the random concurrent execution of a given set of processes $P_1, \ldots, P_k$? Let $S(P_i)$ be the set of possible computation histories of the process P_i. This problem can then be formulated: Given a word w and languages $S(P_1), \ldots, S(P_k)$, is w in the shuffle of the languages $S(P_1), \ldots, S(P_k)$? In [OGD,RID78] it was shown that the shuffle of two context free languages can be NP complete. Using similar techniques ([HAU,WAR80]) one can find a linear language L such that $L \odot L$ is NP complete. Thus the above problem is NP complete for $k = 2$ if the languages $S(P_1)$ and $S(P_2)$ are given as linear grammars, even if these grammars are identical and fixed for all w. On the other hand, if k is fixed and if $S(P_1), \ldots, S(P_k)$ are regular, given as regular grammars or non deterministic finite automata, we can form the nondeterministic finite automaton A for $\bigodot_{i=1}^{k} S(P_i)$ in deterministic polynomial time and check whether or not w is accepted by A via dynamic programming. Hence for any fixed k and $S(P_1), \ldots, S(P_k)$ regular, this problem is solvable in deterministic polynomial time. However, if we allow k to vary, by Theorem 3.1 we see that this problem is NP complete, even when each process is a determined linear sequence of actions, i.e., when $S(P_1), \ldots, S(P_k)$ are single words. This result can be further sharpened using Theorem 3.1. From this theorem we see that the problem of whether or not a given sequence of actions can occur in the random, unbounded concurrent execution of a given determined linear sequence of actions is NP complete. To summarize, the variable language membership problem concerning

computation histories of sets of parallel processes described above becomes difficult if either

1) the processes are allowed to include finite automata augmented with a stack, even if the stack is limited to one reversal or

2) the number of concurrent processes is unbounded.

Results from Section IV and V have some bearing on the complexity of the sets of computation histories resulting from the random concurrent execution of a fixed set of processes. If the sets of computation histories of the individual processes are regular, then the set of computation histories resulting from their random concurrent execution will be regular, since the regular languages are closed under the operation shuffle. However, by the result cited above, if the sets of computation histories of the individual processes are linear, the resulting set of computation histories may be NP complete, even when we consider at most two processes.

If we consider the random, unbounded concurrent execution of a single fixed process, then by Theorem 5.1 we see that the resulting set of computation histories will be recognized in deterministic polynomial time if the set of computation histories of the process is regular. However, since the language $\{a\,b^n\,c\,d\,e^n\,f : n \geq 0\}^{\circledast}$ is NP complete (Theorem 4.2), if we allow the process just a single counter limited to one reversal, the resulting set of computation histories may be NP complete.

It remains to determine the complexity of the variable and fixed language membership problems for the shuffle of a fixed number of single reversal counter languages. However, barring this issue, sharp boundaries have been established with regard to the tractability of the basic variable and fixed language membership problems involving computation histories resulting from random concurrency.

ACKNOWLEDGMENT

We would like to thank Professor Andrzej Ehrenfeucht for numerous helpful discussions concerning this material.

REFERENCES

[ARA,KAG,TOK79] Araki, T., Kagimasa, T. and Tokura, N., Relations of flow languages to Petri Net Languages, Theoretical Computer Science, 15(1)(1981) 51-76.

[GAR,JOH75] Garey, M. and Johnson, D., Complexity results for multi-processor scheduling under resource constraints, SIAM 5, Computing, 4, 397-411, 1975.

[GAR,JOH80] Garey, M. and Johnson, D., Computers and Intractability, A Guide to the Theory of NP Completeness, Problem SP15, p. 224, Freeman Press, 1980.

[GIS79] Gisher, J., Shuffle languages, Petri Nets, and context-sensitive grammars, CACM, 24(9)(1981), 597-605.

[HAU,WAR80] Haussler, D. and Warmuth, M. K., The shuffle of a linear language with itself can be NP complete, unpublished manuscript, 1980.

[JAN79] Jantzen, M., Eigenschaften von Petrinetzsprachen, Doctoral Disseration, Bericht Nr. IFI-HH-B-64, Fachbereich Informatik, Universitat Hamburg, West Germany, 1979.

[JAN81] Jantzen, M., The power of synchronizing operations on strings, Theoretical Computer Science, 14(2)(1981) 127-154.

[KIM76] Kimura, T., An algebraic system for process structuring and interprocess communication, Proc. 8th Annual ACM Symp. on Theory of Computing, pp. 92-100, 1976.

[OGD,RID78] Ogden, W. F., W. E. Riddle, W. C. Rounds, Complexity of expressions allowing concurrency, 5th ACM POPL, Tucson, Arizona, 1978, pp. 185-194.

[RID72] Riddle, W. E., Modelling and analysis of superviosr systems, Ph.D. thesis, Computer Science Dept. Stanford University, 1972.

[SHA75] Shaw, A. C., System design and documentation using path expressions, Proc. Sagamore Computer Conference on Parallel Processing, IEEE Computing Society (1975), pp. 180-181.

[SHA78] Shaw, A. C., Software description with flow expression, IEEE Trans. on Software Engineering 3, SE-4 (1978), pp. 242-254.

[SLU80] Slutzki, G., Descriptional complexity of concurrent processes, unpublished manuscript, Department of Mathematics and Computer Science, Clarkson College of Technology, Potsdam, New York 13676, 1980.